Video games sales

)Hokage Heroes(

TABLE OF CONTENTS.

OUR Dataset

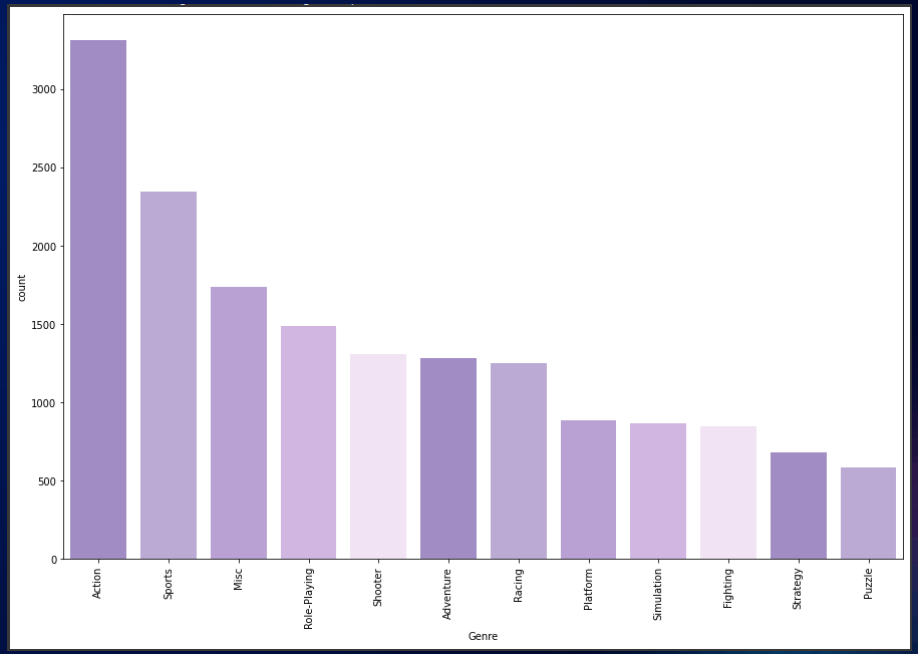
our dataset contains a list of video games with sales greater than 100,000 copies.

In this Presentation, we will be analyzing, processing, and exploring a large amount of data on video game sales. The dataset contains information regarding the sales of video games across various regions like North America, Europe, Japan, and also globally, while also giving information regarding the Names, Publishers, and Platforms .

What genre games have been made the most?

The top 4 genre games that have been made are:

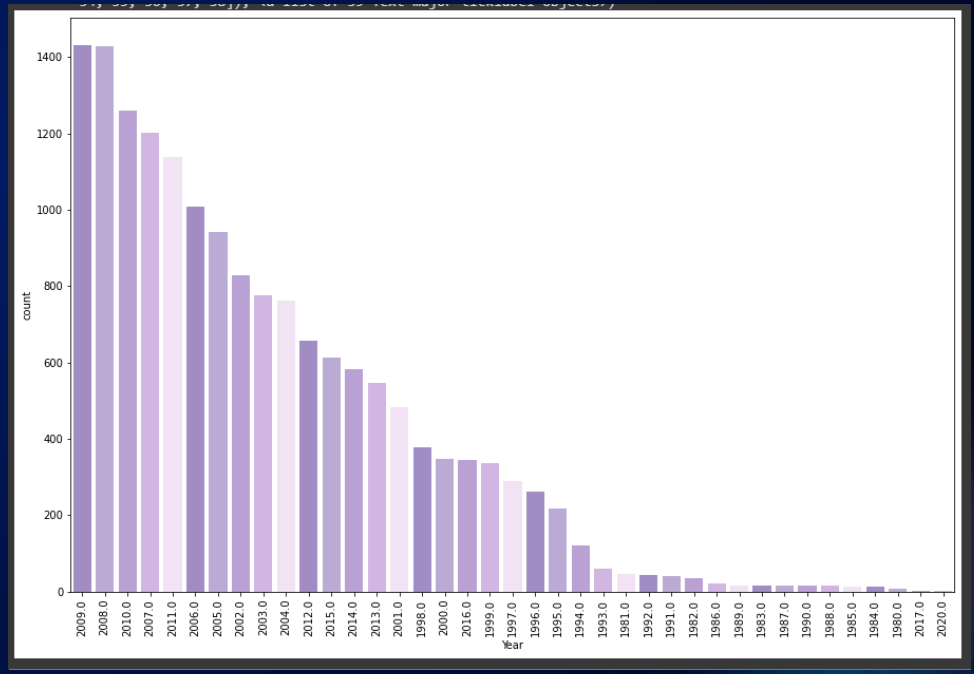
1. Action
2. Sports
3. Misc
4. Role-Playing



what year were the most games released?

The years were the most games were released are:

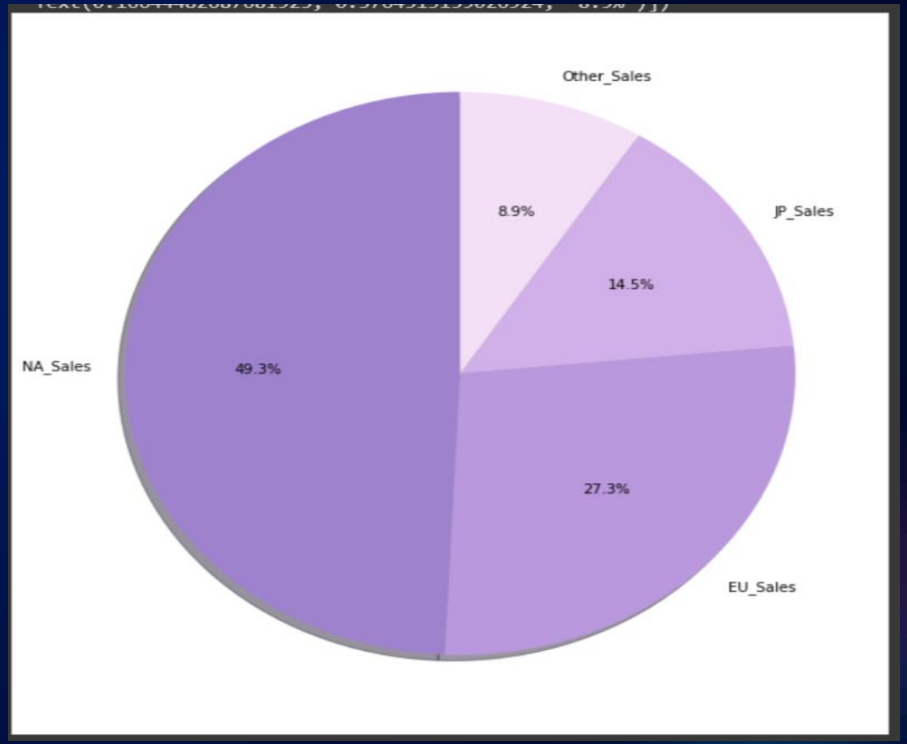
- 2009
- 2008
- 2010



what are total sales by region ?

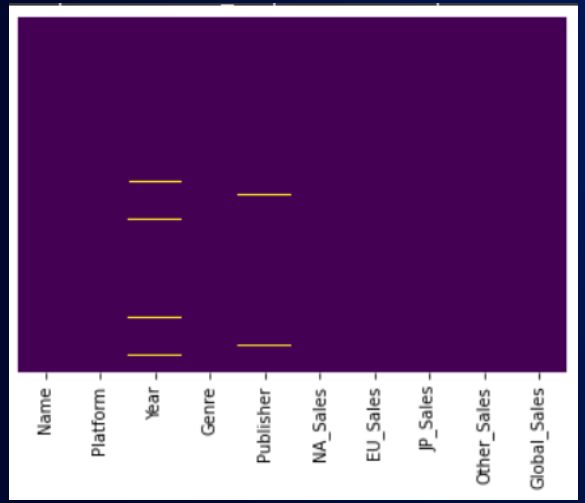
the total sales by region (descending)

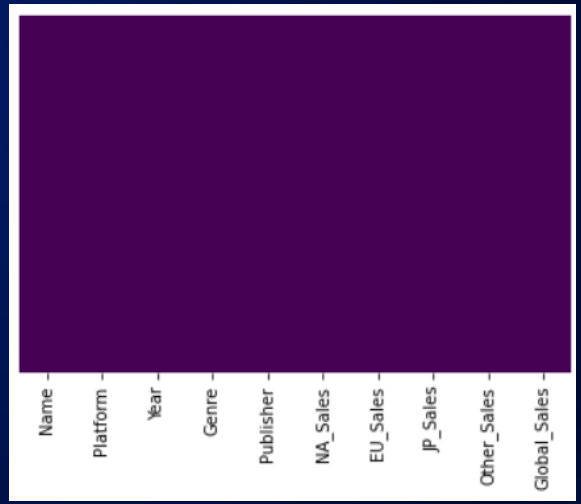
1. NA_Sales(49.3%)
2. EU Sales(27.3%)
3. JP_Sales(14.5%)
4. Other_Sales(8.9%)



Data processing

- Null Values





Data processing

- Convert Float to int

```
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Rank          16598 non-null  int64
 1   Name          16598 non-null  object
 2   Platform      16598 non-null  object
 3   Year          16327 non-null  float64
 4   Genre         16598 non-null  object
 5   Publisher     16540 non-null  object
 6   NA_Sales      16598 non-null  float64
 7   EU_Sales      16598 non-null  float64
 8   JP_Sales      16598 non-null  float64
 9   Other_Sales   16598 non-null  float64
 10  Global_Sales  16598 non-null  float64
```

```
 #   Column        Non-Null Count  Dtype
---  ------        --------------  -----
 0   Name          16291 non-null  object
 1   Platform      16291 non-null  object
 2   Year          16291 non-null  int64
 3   Genre         16291 non-null  object
 4   Publisher     16291 non-null  object
 5   NA_Sales      16291 non-null  int64
 6   EU_Sales      16291 non-null  int64
 7   JP_Sales      16291 non-null  int64
 8   Other_Sales   16291 non-null  int64
 9   Global_Sales  16291 non-null  int64
dtypes: int64(6), object(4)
memory usage: 1.4+ MB
```

Data processing

- Encoding object to int

```
0   Name          16291 non-null  object
1   Platform      16291 non-null  object
2   Year          16291 non-null  int64
3   Genre         16291 non-null  object
4   Publisher     16291 non-null  object
```

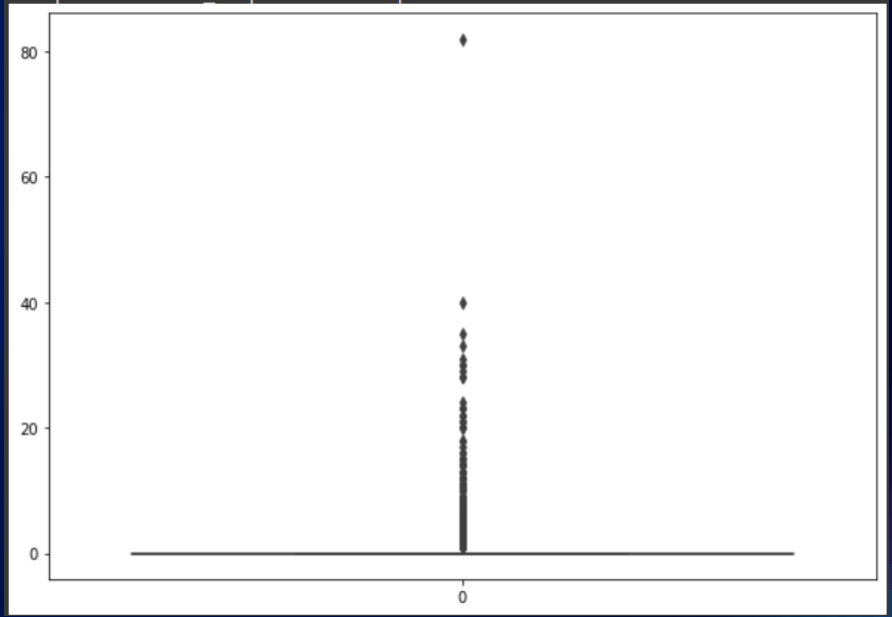
```
10  cat-Platform   16291 non-null  int8
11  cat-Genre      16291 non-null  int8
12  cat-Publisher  16291 non-null  int16
```

Data processing

- Outlier
  - Global sales > 80

```
# outlire Global_Sales
plt.figure(figsize=(10,7))
sns.boxplot(data=df['Global_Sales'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f50c80015d0>
```

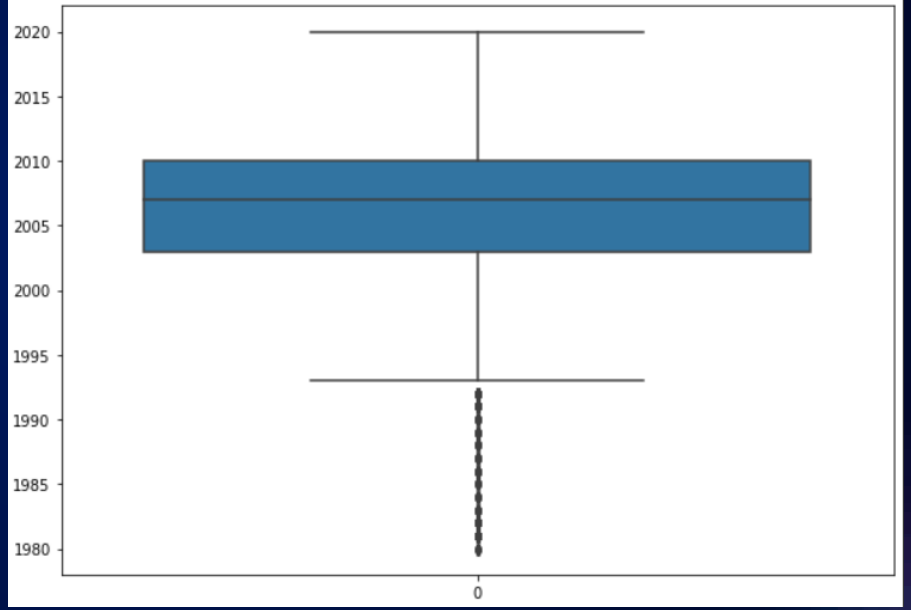


Data processing

- Outlier
  - Global sales > 80
  - Year < 1990

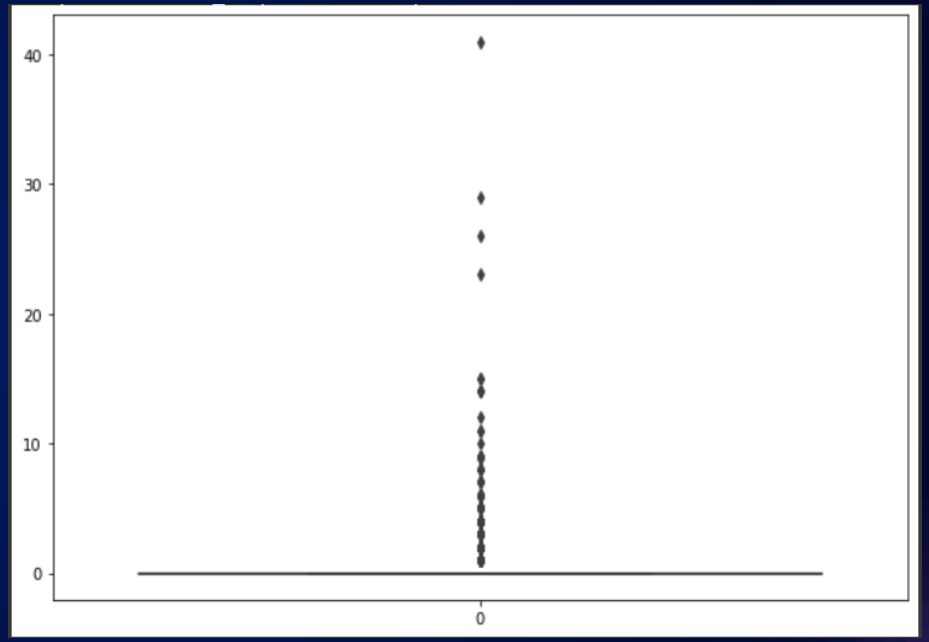
```
sns.boxplot(data=df['Year'])
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f50c7986310>
```



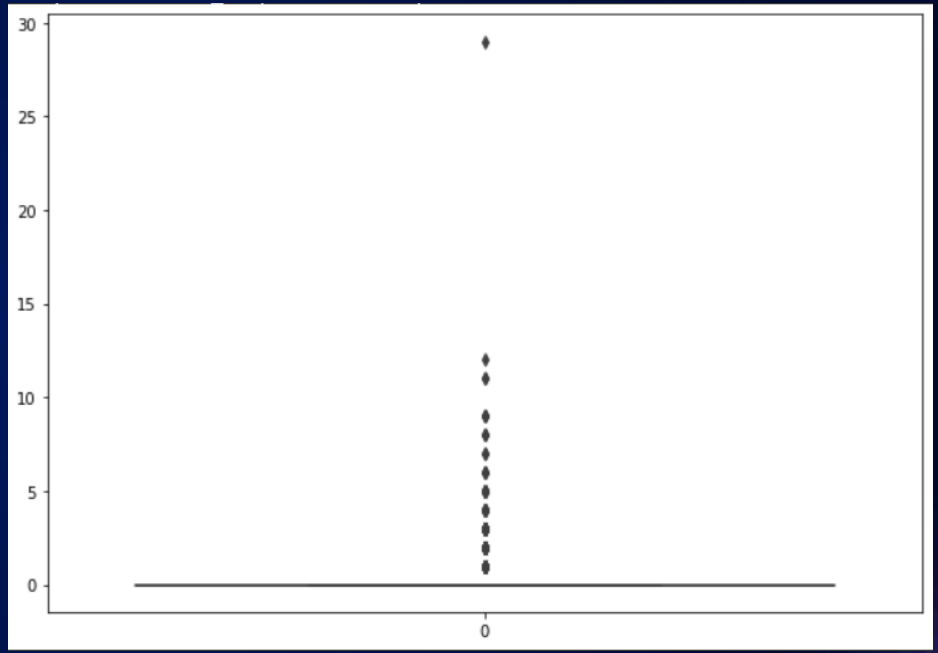
Data processing

- Outlier
  - Global Sales > 80
  - Year < 1990
  - NA Sales > 20



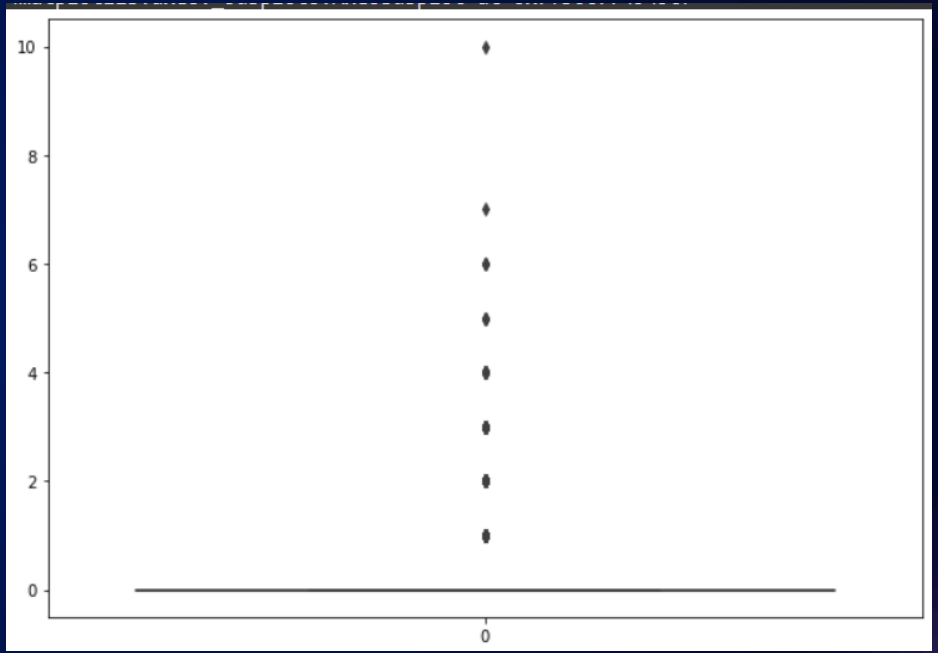
Data processing

- Outlier
  - Global Sales > 80
  - Year < 1990
  - NA Sales > 20
  - EU Sales > 25



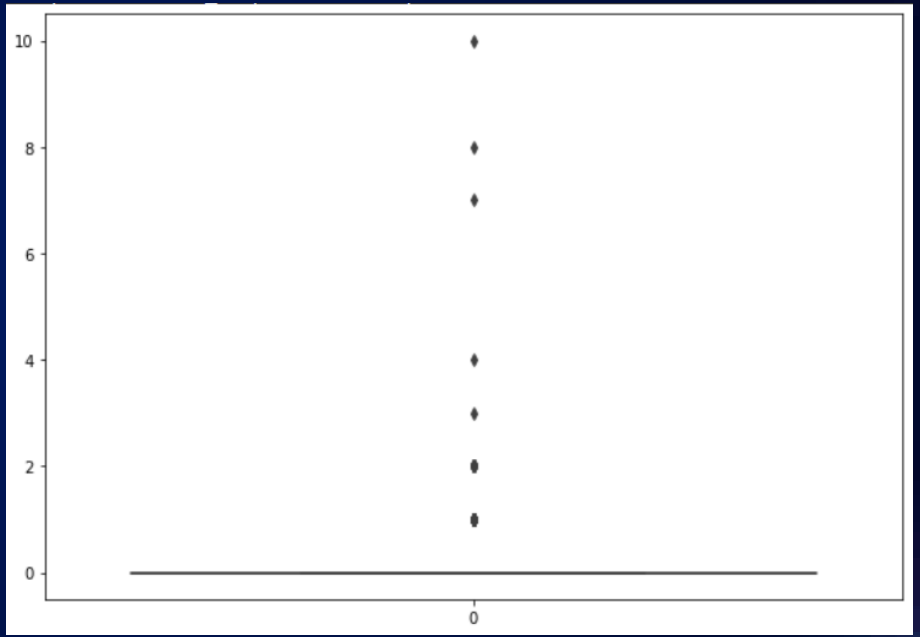
Data processing

- Outlier
  - Global Sales > 80
  - Year < 1990
  - NA Sales > 20
  - EU Sales > 25
  - JP Sales > 8



Data processing

- Outlier
  - Global Sales > 80
  - Year < 1990
  - NA Sales > 20
  - EU Sales > 25
  - JP Sales > 8
  - Other Sales > 6



Regression models

simple linear regression

```
# Split the data
train, test = train_test_split(
    df,
    test_size=0.2,
    train_size = 0.8,
    random_state = 42
)
```

```
# Create my X, y data

target = "Global_Sales"
features = "NA_Sales"

X_train = train[[features]]
y_train = train[[target]]

X_test = test[[features]]
y_test = test[[target]]
```

```

# Create a model object
gm = LinearRegression()

# Train the model
gm.fit(X_train, y_train)
```

```
r2_score(y_true=y_test, y_pred=predictions)
```

```
0.8513438991913049
```

```
mean_absolute_error(y_true=y_test, y_pred=predictions)
```

```
0.2125363539133119
```

Multiple linear regression

```
[ ]
target = "Global_Sales"
features = ['Year','NA_Sales','EU_Sales','JP_Sales','Other_Sales','cat-Platform','cat-Genre','cat-Publisher']

X_train = train[features]
y_train = train[[target]]

X_test = test[features]
y_test = test[[target]]
```

```
[ ]

# Create a model object
multi_gm= LinearRegression()

# Train the model
multi_gm.fit(X_train, y_train)
```

```
LinearRegression()
```

```
[ ] multi_gm.intercept_
```

```
array([2.58568975])
```

```
[ ] multi_gm.coef_
```

```
array([[-1.25921832e-03,  1.18262168e+00,  1.19350189e+00,
         1.02981862e+00,  5.76697402e-01,  1.02161469e-03,
        -1.10817281e-03,  3.42727448e-05]])
```

```
[ ] multi_predictions = multi_gm.predict(X_test)
```

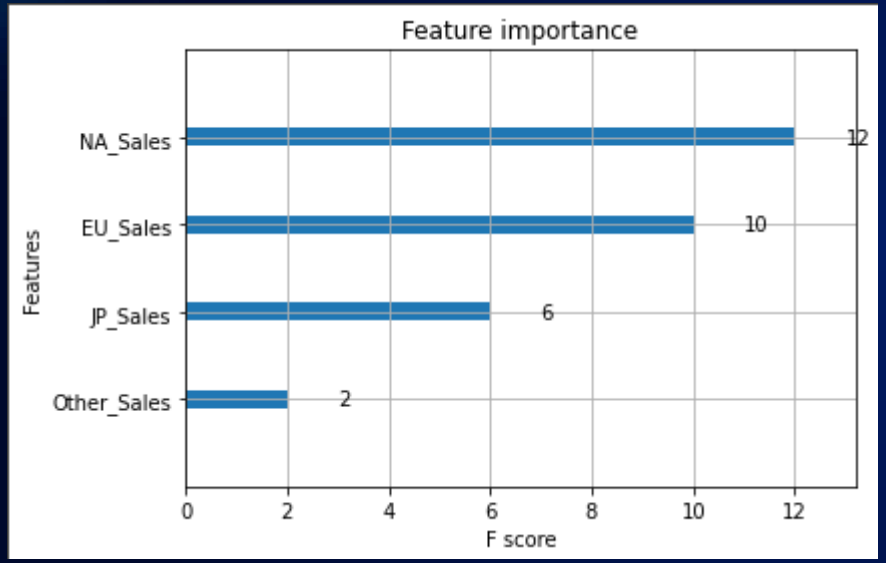
```
[ ] r2_score(y_true=y_test, y_pred=multi_predictions)
```

```
0.9519540857305402
```

```
mean_absolute_error(y_true=y_test, y_pred=multi_predictions)
```

```
0.1591725296429255
```

Regression using XGBoost model

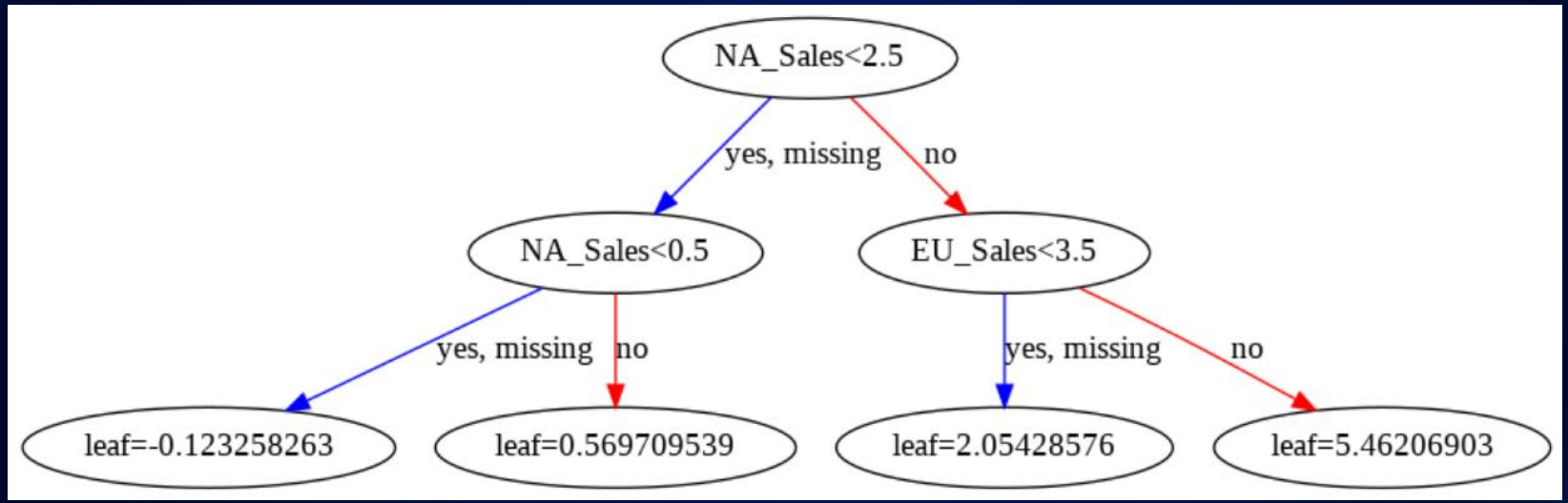



```
[183] # Computing RMSE
      print("RMSE: %f" % (np.sqrt(mean_squared_error(y_test, predictions))))

      RMSE: 0.302012

[184] # Computing MAE
      print("MAE: %f" % (mean_absolute_error(y_test, predictions)))

      MAE: 0.143691
```



classification models

Create Sales Column

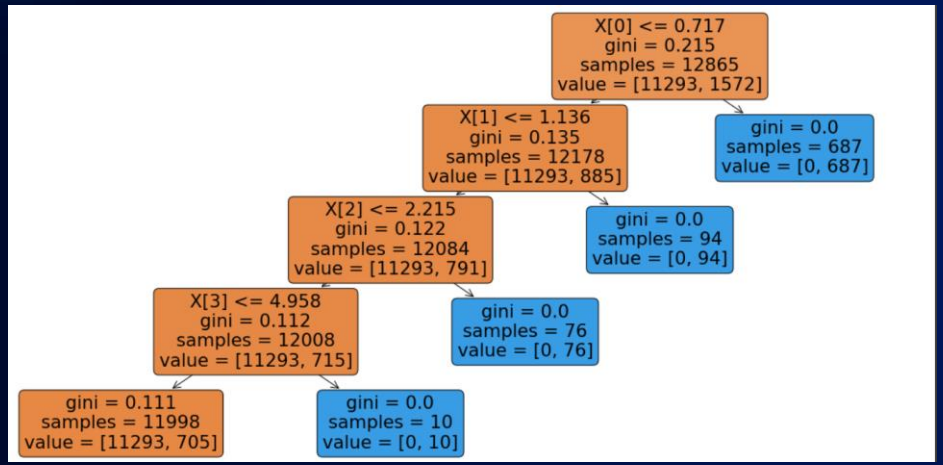
```
df['Sales'] = True
for i in df.index:
  if df['Global_Sales'][i] == 0:
    df['Sales'][i] = False
```

Classification Models: Logistic Regression

	Actual_Purchased	Predict_Purchased
0	False	False
1	False	False
2	True	True
3	False	False
4	False	False

	precision	recall	f1-score
True	0.95	1.00	0.97
False	1.00	0.58	0.74
accuracy			0.95
macro avg	0.97	0.79	0.85
weighted avg	0.95	0.95	0.95

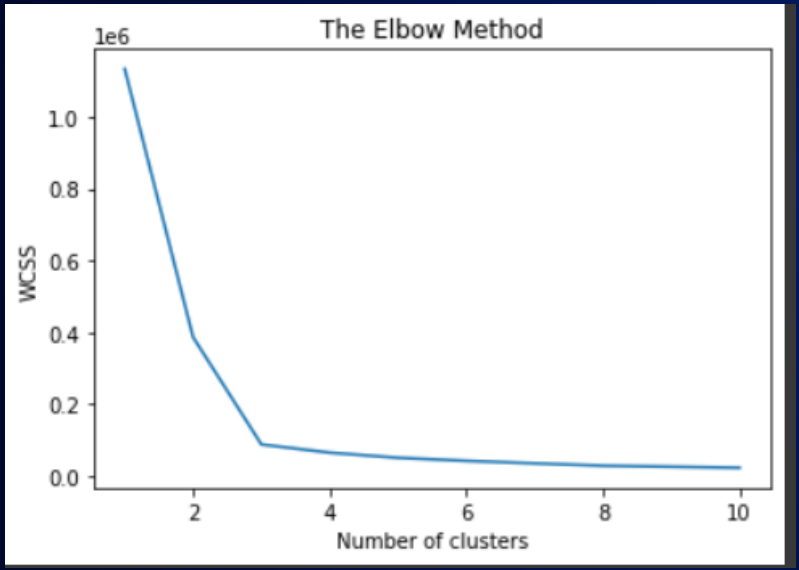
Decision Tree Classification Model Using Gini Index

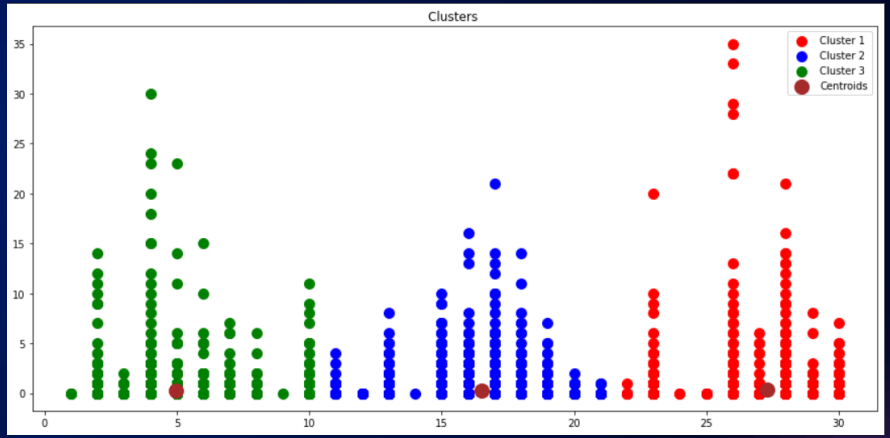



```
print(f'Training Accuracy: {val_train}%')
print(f'Test Set Accuracy: {val_test}%')
```

```
Training Accuracy: 95.0%
Test Set Accuracy: 95.0%
```

K-Nearest Neighbor(KNN)





PipeLine

```
clf = Pipeline(
    steps=[
        ('preprocessor', preprocessor),
        ('classifier', DecisionTreeClassifier())
    ]
)

clf.fit(X_train, y_train)

print(f"model score: {clf.score(X_test, y_test)}")
```

```
model score: 1.0
```

Result

THANK YOU!

Do you have any questions?

Hayam Alrashed
Shouq Alharbi
Sarah Alrashidi
Razan Alajlan
Nada Oteif